Fast and Dangerous

The Misunderstood Oracle ROWID

By Bert Scalzo, Ph.D.

The Oracle ROWID is an important and powerful SQL programming tool. However, ROWIDs have a history of being somewhat misunderstood, especially regarding when and how they can safely be used.

Under the right circumstances, ROWIDs can make your SQL and/or PL/SQL code run faster.

But beware! ROWIDs also have the ability to make your program behave erratically — or not at all. Therefore, the ROWID must be treated with great respect.

Just What Is It Exactly?

What is an Oracle ROWID? Every row in an Oracle table is assigned a physical address known as its ROWID. For non-clustered tables, these addresses are unique. For clustered tables, rows that are from different tables, but are located within the same data block, will have identical ROWIDs.

While one may project and restrict using the ROWID much like any other column in a table, it is nonetheless a pseudo-column. Oracle pseudo-columns are not actually part of the table, and as such do not show up in a table describe. Nonetheless, they are quite useful.

Other examples of pseudo-columns include CURRVAL and NEXTVAL (sequence number generators), LEVEL (for hierarchical queries), and ROWNUM (a row's relative number).

Internally, ROWID is a binary value. Externally, however, it is displayed as a three-part hexadecimal string stored in a VARCHAR2 column.

The first part (positions 1 through 8) is the block ID; the second part (positions 10 through 13) is the sequence number of the row within the block; and the third part (positions 15 through 18) is the data file ID.

You can separate ROWID into its components using the SUBSTR function.

This SELECT statement for example:

```
SELECT ROWID,
       SUBSTR(ROWID, 15,  4)   AS "FILE",
       SUBSTR(ROWID,  1,  8)   AS "BLOCK",
       SUBSTR(ROWID, 10,  4)   AS "ROW"
  FROM dual;
```

might yield:

```
ROWIDFILEBLOCKROW
------------------------------------------
0000033D.0000.0001000100000033D0000
```

Or, easier to read:

ROWID	0000033D.0000.0001
File	0001
Block	0000033D
Row	0000

In this example, the row is the first row within its block (because its row offset within the block is zero). Moreover, it is

located in the 829 block within data file one. This is the row's physical address.

It's *Fast!*

Accessing a table's rows using the ROWID is fast. In fact, the rule-based optimizer considers access by ROWID as the highest ranked access method.

The rules for access path selection, according to the *Oracle Application Developers Guide*, are listed in Figure 1.

Rank	Access Path
1	Single row by ROWID
2	Single row by cluster join
3	Single row by hash cluster key with unique or primary key
4	Single row by unique or primary key
5	Cluster join
6	Hash cluster key
7	Indexed cluster key
8	Composite key
9	Single-column indexes
10	Bounded range search on indexed columns
11	Unbounded range search on indexed columns
12	Sort-merge join
13	MAX or MIN of indexed column
14	ORDER BY on indexed columns
15	Full table scan

Figure 1: The rules for access path selection.

As you can see, the first — and therefore fastest — access method is the ROWID. This access path will only be available if:

- the statement's WHERE clause identifies the selected rows by ROWID,
- or with the CURRENT OF CURSOR clause.

If one of these criterion is met, the Oracle server can access the data rows via their physical address. The result is lightning-fast access.

This ultra-fast access technique has an Achilles heal, however: *ROWIDs can change.*

Ch... Ch... Changes

The address of a row can change under different circumstances. For example, an export followed by an import will usually result in different ROWIDs. However, this is an innocuous example. Because the ROWID is changing outside the realm of the user's transactions, there is no reason for concern.

Contrast this to a ROWID changing during the processing of one's DML operations. If you have relied on a ROWID being consistent for the entire scope of your program, you may be in real trouble:

```
PROCEDURE transfer_all
   (commit_after_in IN INTEGER   := 100,
    requery_after_in IN INTEGER  := 1000)
IS
   more_data BOOLEAN     := TRUE;
   commit_count INTEGER  := 0;

   CURSOR trans_cur
   IS
      SELECT ROWID FROM transfer_table
       WHERE transfer_indictor = 'U'
         AND ROWNUM <= requery_after_in;
BEGIN
   WHILE more_data
   LOOP
      more_data := FALSE;
      FOR trans_rec IN trans_cur
      LOOP
         more_data := TRUE;

         INSERT  INTO master_table
         SELECT  *
           FROM  transfer_table
          WHERE  ROWID = trans_rec.ROWID;

         UPDATE  transfer_table
            SET  transfer_indicator = 'T'
          WHERE  ROWID = trans_rec.ROWID;

         commit_count := commit_count + 1;
         IF commit_count = commit_after_in
         THEN
            COMMIT;
            commit_count := 0;
         END IF;

      END LOOP;
   END LOOP;
END;
```

Figure 2: The transfer_all procedure.

ROWIDs are only guaranteed to be consistent during the length of a transaction.

An Oracle transaction starts the moment you connect until either a COMMIT or ROLLBACK occurs, at which time another transaction begins. Hence, ROWIDs may change within the scope of your program if you have more than one transaction.

An Example

A perfect example of this problem is the transfer_all procedure (see Figure 2) that is the solution to April's PL/SQL challenge (see page 63 of the May 1996 issue of *Oracle Informant*). This PL/SQL procedure constructs a cursor, TRANS_CUR, from selected ROWIDs that match some restriction criteria.

Then, the code has a cursor FOR loop that steps through those returned rows. The loop includes both INSERT and UPDATE DML commands that reference the ROWID of the current cursor record.

Also inside this cursor FOR loop is a COMMIT statement that is executed at every threshold interval as indicated by the `commit-count` variable. When a

COMMIT occurs, the ROWIDs held by the cursor are free to change.

Still, no problem is likely to occur in a single-user environment. But in a multi-user, concurrent transaction environment, this code could very easily have problems:

- Another user could perform UPDATEs, INSERTs, or DELETEs that would invalidate your ROWIDs.
- In addition, free space compression which occurs internally and automatically during INSERTs, and UPDATEs that decrease lengths, can also alter ROWIDs.

This problem might never manifest itself. More likely, however, it would be one of those weird, nagging errors that is hard to reproduce, and nearly impossible to debug.

Conclusion

ROWIDs should be used sparingly, and under stringent guidelines. My rule is that one should never use a ROWID across SQL statements without locking those rows.

The *Oracle7 Server Concept Manual* asserts "Before ROWIDs are used in DML statements, they should be verified and guaranteed not to change; the intended rows should be locked so they cannot change."

And *Oracle Performance Tuning* [Corrigan and Gurry, O'Reilly & Associates, 1993] states "Remember when you first query the record, to select the record for update. This keeps another process from being able to update the selected record and change its ROWID out from under you."

In the case of the transfer_all procedure, two changes would be necessary to fix the PL/SQL code. First, add:

```
FOR UPDATE OF ROWID
```

to the cursor declaration. Second, move the COMMIT statement outside the FOR loop. While this may seem to make the program less flexible, it nonetheless makes it entirely reliable.

Bert Scalzo is a senior consultant for Logic Works Inc. He provides consulting and education on data modeling and Oracle database design. He has his Ph.D., Masters, and Bachelors degrees in Computer Science, plus an MBA and several insurance designations. Moreover, Dr Scalzo has numerous Oracle Masters certifications and has worked for both Oracle Education and Oracle Consulting. He can be reached via the Internet at bscalzo@logicworks.com.

Available December 1996

The Must Have Reference Source For The Serious Oracle® Developer

oracle informant

CD ROM

The Entire Text of all Technical Articles Appearing in Oracle® Informant® in 1996

The Oracle® Informant® Works 1996 CD-ROM will include:

- All Technical Articles
- Text and Keyword Search Capability
- Improved Speed and Performance
- All Supporting Code and Sample Files
- 16-Page Color Booklet
- Third-Party Add-In Product Demos
- CompuServe Starter Kit with $15 Usage Credit.

A $130 Value
Available Now for only

$49.95

California residents add 7¼% Sales Tax, plus $5 shipping & handling for US orders.
(International orders add $15 shipping & handling)

Call Now Toll Free
1-800-88-INFORM

1-800-884-6367 Ask for offer # WEB
To order by mail,
send check or Money Order to:
Informant Communications Group, Inc.
ATTN: Works CD offer # WEB
10519 E. Stockton Blvd, Suite 142
Elk Grove, CA 95624-9704
or Fax your order to 916-686-8497

Get Informed!

Subscribe to Oracle Informant, The Independent Monthly Guide to Oracle Development.

Order Now and Get One Issue FREE!

For a limited time you can receive the first issue FREE plus 12 additional issues for only $49.95 That's nearly 25% off the yearly cover price!

Each big issue of *Oracle Informant* is packed with Oracle tips, techniques , news, and more!

- Client/Server Development
- Using Developer/2000™
- Tuning Oracle7
- PL/SQL Techniques
- Advanced Oracle Topics
- Distributed Managment
- Product Reviews
- Book Reviews
- News from the Oracle Community
- Oracle User Group Information

Order Now!

YES!, I want to sharpen my Oracle skills. I've checked the subscription plan I'm interested in below

☐ Magazine-Only Subscription Plan...
13 Issues Including One Bonus Issue at $49.95.

☐ Magazine AND Companion Disk Subscription Plan...
13 Issues and Disks Including One Bonus Issue and Disk at $119.95
The Oracle Informant Companion Disk contains source code, support files, examples, utilities, samples, and more!

☐ Oracle Informant Works 1996 CD-ROM $49.95 (Available December 1996)
US residents add $5 shipping and handling. International customers add $15 shipping and handling.

Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

Country _____ Phone _____

FAX _____ E-Mail _____

Payment Method...

❑ Check (payable to Informant Communications Group) ❑ Purchase Order-- Provide Number _____

❑ Visa ❑ Mastercard ❑ American Express Card Number _____

Expiration Date _____ Signature _____

WEB

To order, mail or fax the adjoining form or call (916) 686-6610 Fax: (916) 686-8497

International rates

Magazine-only
$54.95/year to Canada
$74.95/year to Mexico
$79.95/year to all other countries

Magazine AND Companion Disk Subscriptions
$124.95/year to Canada
$154.95/year to Mexico
$179.95 to all other countries

California Residents add 7¼% sales tax on disk subscription